**Trabajo Práctico 2**

1. ¿Qué es la herencia en java? ¿Cuáles son sus beneficios?

La herencia es un procedimiento a través del cual declarando una clase padre y otra clase hija, la clase hija puede hacer uso de todo lo que se encuentra en la clase padre, es decir, sus atributos y métodos.

Los beneficios son que nos permite reutilizar código y que hace que el código sea mas legible y de fácil mantenimiento.

1. ¿Cómo se representa la herencia en un diagrama UML?

En un diagrama UML la herencia se representa con una flecha que inicia en la clase hija, y que tiene la punta en la clase padre. “La clase hija hereda de la clase padre”

1. ¿Cuándo es recomendable usar la herencia?

Es recomendable usar la herencia cuando tenemos dos clases que tienen cosas en común, una de ellas es general y la otra comparte todo lo general, pero es una versión más específica.

1. ¿Qué es una interfaz en java?

Una interfaz es una colección de métodos abstractos que se definen pero no se implementan. Permiten la implementación de múltiples herencias.

1. ¿Cómo se representa una interfaz en un diagrama UML?

La interfaz se representa con una línea punteada y una flecha en la punta. Además, se declara arriba del rectángulo que se está representando una interfaz: <<interface>>

1. ¿Cuándo es recomendable usar una interfaz?

Cuando necesitamos hacer herencia multiple, y cuando cada clase que implementa esa interfaz necesita una implementación especifica de cada método.

1. ¿Cuál es la diferencia entre una interfaz y una clase abstracta?

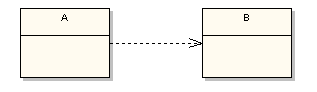
En la interfaz pueden declararse constantes y métodos no implementados. En cambio, en la clase abstracta pueden declararse contantes, pero también atributos y los métodos pueden estar implementados o no, y también pueden ser abstractos.

1. ¿Qué tipos de relaciones pueden existir entre clases y objetos?

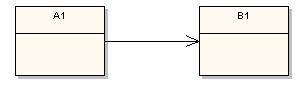
Dependencia, asociación, agregación, composición y generalización.

1. ¿Cómo se representa cada relación en un diagrama UML?

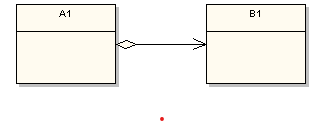
Dependencia: La clase A tiene algún tipo de dependencia de la clase B.



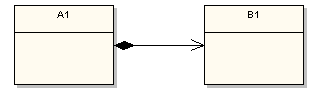
Asociación: La punta de flecha significa que existe una relación de un solo sentido. La clase A1 utiliza y contiene una instancia de la clase B1, pero B1 no sabe nada de que contiene instancias en la clase A1.



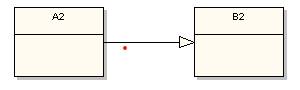
Agregación: A1 tiene una instancia de B1 pero B1 es independiente



Composición: A1 tiene una instancia de B1 y B1 es independiente



Generalización: A2 hereda de B2



1. ¿Cuándo es recomendable usar cada tipo de relación?

Dependencia: Es preferible no usarla porque representa que una clase depende de otra clase en algún aspecto y este aspecto puede ser mucho mas especifico usando alguna de las otras relaciones.

Asociación: Cuando dos clases están relacionadas, pero no hay una dependencia fuerte entre ellas.

Agregación: Cuando una clase contiene a otra clase como parte de sí misma, pero la otra clase puede existir independientemente.

Composición: Cuando una clase contiene a otra clase como parte de sí misma y la otra clase no puede existir independientemente.

Generalización: Cuando necesitamos representar una herencia entre clases.